Remarks/Arguments:

Claims 1-12 have been amended. Claim 13 has been added. No new matter is introduced herein. Claims 1-13 are pending.

Claim 1 has been amended to clarify that the method is a computer system implemented method of automatically computing from data received by a computer-based value appraising system an optimal appraisal value of a real estate property. Claim 1 has further been amended to clarify that steps (a), (b) and (d) are performed by the computer-based value appraising system. Finally, claim 1 has been amended to recite a step of outputting the optimal appraisal value to the user. Claims 2-11 have been amended to correspond with claim 1. Claim 12 has been amended to include computer-executable instructions, at least one data processor that executes the instructions and to recite "a computer-readable medium comprising computer-executable code for causing a processor to perform." No new matter is introduced herein. Support for the amendments include, for example, page 5, line 18-page 6, line 11; page 11, line 10-page 12, line 11; and page 15, line 18-page 18, line 18 of the substitute specification filed on October 14, 2003.

Claims 1-12 have been rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Claim 1 has been amended, as discussed above, to recite a computer system implemented method and that steps (a), (b) and (d) are performed by a computer-based value appraising system. Claim 12 has been amended to include computer-executable instructions and "a computer-readable medium comprising computer-executable code for causing a processor to perform."

On page 3 of the Office Action (relating to the § 101 rejection), the Examiner asserts that "Applicant's claimed invention is directed to a human calculating appraisal of a real estate [by] programming a programmable calculator with a nonlinear function, entering the data and generating an appraised value." Applicant respectfully disagrees. Applicant's claims 1 and 12 are directed to <u>nonlinear programming of a nonlinear objective function</u> to simultaneously optimize the nonlinear objective function for all of different types of appraisal approaches. As known to the skilled person, nonlinear programming is a complicated technique that attempts to minimize or maximize a function of several variables (i.e., an objective function) to determine an optimum solution to the objective function. The skilled person understands that, because of the complexity of nonlinear programming techniques, <u>computers are typically used to implement nonlinear programming</u>. For the

Examiner's convenience, a copy of U.S. 2001/0051936 to Michalewicz, which relates to nonlinear programming, is attached. Thus, the skilled person, upon reading Applicant's specification, would understand that Applicant's invention relates to a computer-based value appraising system and uses a processor to perform the nonlinear programming of the nonlinear objective function, as recited in Applicant's claims 1 and 12. Accordingly, Applicant respectfully requests that the rejection of claims 1-12 under 35 U.S.C. § 101 be withdrawn.

Claims 1-12 have been rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement. In particular, the Examiner asserts that the "Specification, while being enabling for a human to perform nonlinear programming of a microprocessor ... does not reasonably provide enablement for microprocessor to perform the nonlinear programming." Applicant respectfully disagrees. Support for a processor performing the nonlinear programming is provided in the substitute specification at page 5, line 18-page 6, line 11; page 11, line 10-page 12, line 11; page 15, line 18-page 18, line 18. Applicant also notes that, as discussed above, the skilled person understands that nonlinear programming is typically performed by a microprocessor, rather than by a human. Applicant again respectfully points the Examiner's attention to the attached Michalewicz reference. Accordingly, for the reasons set forth above, Applicant respectfully requests that the rejection of claims 1-12 under 35 U.S.C. § 112, first paragraph, be withdrawn.

Claims 1-12 have been rejected under 35 U.S.C. § 112, second paragraph, as being unclear. In particular, it is asserted, on page 4 of the Office Action, that "it is not clear whether a human performs the programming of the microprocessor, or, microprocessor performs the programming of itself" and that Applicant "has not positively claimed if any action is performed with the generated appraisal of the real estate." As discussed above, claim 1 has been amended to clarify that the method relates to a computer system implemented method and that steps (a), (b) and (d) are performed by a computer-based value appraising system. Claim 1 has also been amended to clarify that the optimal appraisal value is output to the user. Claim 12 has been amended to include computer-executable instructions and "a computer-readable medium comprising computer-executable code for causing a processor to perform." Accordingly, Applicant respectfully requests that the rejection of claims 1-12 under 35 U.S.C. § 112, second paragraph be withdrawn.

Claims 1-12 have been rejected under 35 U. S. C. § 103(a) as being unpatentable over Robbins (US 2001/0039506) in view of "Modern Real Estate Practice" by Galaty et al.

and Bradley et al. (U.S. 7,711,574). Reconsideration is respectfully requested for the reasons set forth below.

Claim 1 includes features neither disclosed nor suggested by the cited art, namely:

> ...where all of different types of appraisal approaches are used together to optimize a nonlinear objective function...
>
> b) defining, by the computer-based value appraising system, the nonlinear objective function that includes control variables representing the stored influence factors for all of the different types of appraisal approaches;
>
> c) using and causing a microprocessor to perform nonlinear programming of the nonlinear objective function to simultaneously optimize the nonlinear objective function for all of the different types of appraisal approaches by adjusting the control variables within the corresponding range of influence factor values... (Emphasis Added)

Claim 12 includes a similar recitation.

Robbins teaches the determination of a real estate parcels market value through the application of the sales comparison approach to value (paragraph [0076]). In Figs. 3 and 4, Robbins teaches that a set of procedures are created to build property attribute databases and a set of procedures are created to apply the rules of appraisal to the property attribute databases in order to estimate the value of a subject property ([0105]). In Figs. 4 and 5, Robbins teaches the development of a sales condition score for individual parcels that may be used to suggest to the user a relationship between a comparable selling price and its attribute inventory (paragraphs [0133 - 0144]). Robbins also teaches that the invention assists in the reliability of the sales comparison approach by providing access to an increased number of substitute properties (paragraph [0080]).

Robbins, however, does not teach: 1) defining a nonlinear objective function that includes control variables representing the stored influence factors for all of the different types of appraisal approaches, 2) nonlinear programming to simultaneously optimize the nonlinear objective function for all of the different types of appraisal approaches or 3) that all of the different types of appraisal approaches are used together to optimize the nonlinear objective function, as required by claims 1 and 12. Robbins is silent regarding these indicated features.

Robbins merely discloses, at paragraph [0080], that "an appraiser generally considers three separate approaches to value: the cost approach, the income approach, and the sales comparison approach" but that "the invention is specific to the sales comparison approach" (emphasis added). In other words, Robbins mentions that, in determining the market value of a subject property, an appraiser generally considers three separate appraisal approaches to value. However, Robbins does not disclose or suggest using these approaches together., but that the appraiser considers the appropriateness of the approaches to value in order to select the most appropriate approach (paragraph [0081]). Thus, Robbins does not include all of the features of claims 1 and 12.

Galaty et al. disclose that appraisers traditionally use the sales comparison approach, the cost approach and the income approach, where the three methods serve as checks against each other (p. 304, last paragraph). At p. 305-p. 312, Galaty et al. disclose linear calculations for separately appraising value by each of the three methods. At p. 312, Galaty et al. teach that when the three approaches are applied to the same property, three separate indications of value are produced. Reconciliation (based on the validity of each approach) is then performed to produce a single estimate of the market value. Thus, as shown in Fig. 18.1, separate calculations for each of the different appraisal value approaches are performed and reconciled (i.e., after they are separately calculated) to generate a single estimate of a market value.

Galaty et al., however, do not disclose or suggest: 1) defining a nonlinear objective function that includes control variables representing the stored influence factors for all of the different types of appraisal approaches, 2) nonlinear programming to simultaneously optimize the nonlinear objective function for all of the different types of appraisal approaches or 3) that all of the different types of appraisal approaches are used together to optimize the nonlinear objective function, as required by claims 1 and 12. Galaty et al. are silent regarding these indicated features.

At page 304, Galaty et al. only mention three separate appraisal approaches to value "as checks against each other" for narrowing "the range within which the final estimate of value falls." As discussed above, Galaty et al. only teach separately calculating the property value, followed by a reconciliation process to produce a single estimate of the market value. Thus, Galaty et al. do not make up for the deficiencies of Robbins with respect to claims 1 and 12.

Bradley et al. relate to systems and methods to determine an indication (such as a home value (HV) score) of whether a property appraisal is likely to be faulty. (Column 4, lines 6-22.) Bradley et al. disclose, in Fig. 12, a logistic regression approach to determine coefficients for the HV score model, based on historical information and verification of a first appraisal. Logistic regression models are used to examine how various factors influence a binary outcome. (Column 14, line 64-column 16, line 64.)

Bradley et al., however, do not disclose or suggest: 1) defining a nonlinear objective function that includes control variables representing the stored influence factors for all of the different types of appraisal approaches, 2) nonlinear programming to simultaneously optimize the nonlinear objective function for all of the different types of appraisal approaches or 3) that all of the different types of appraisal approaches are used together to optimize the nonlinear objective function, as required by claims 1 and 12. Bradley et al. are silent regarding these indicated features.

On page 6 of the Office Action, the Examiner asserts that "Robbins in view of Galaty does not teach what formula(e) it uses to determine appraised value of a real estate." Applicant respectfully disagrees. Galaty et al. provide specific examples and specific formulae for the sales comparison approach, the cost approach and the income approach. (See pages 305-312 of Galaty et al.) Bradley et al. describe logistic regression techniques to determine coefficients of an HV score model. However, none of the cited art disclose or suggest nonlinear programming and a nonlinear objective function. As described in Applicant's specification, and as known to the skilled person, nonlinear programming is a specific technique which maximizes or minimizes an objective function. These indicated features are completely absent from the cited art.

Accordingly, none of the cited art, either alone or in combination, can teach: 1) defining a nonlinear objective function for all of the different types of appraisal approaches, 2) nonlinear programming to simultaneously optimize the nonlinear objective function for all of the different types of appraisal approaches or 3) that all of the different types of appraisal approaches are used together to optimize the nonlinear objective function, as required by claims 1 and 12. Thus, for the reasons set forth above, allowance of claims 1 and 12 is respectfully requested.

Claims 2-11, which include all of the features of claim 1, are also patentable over the cited art for at least the same reasons as claim 1.

Claim 13 has been added. No new matter is introduced herein. Claim 13 is similar to claim 12. Claim 12 recites a processor coupled to at least one memory, where the "processor is programmed to perform the optimal appraisal by." Applicant also points the Examiner's attention to the Interim Examination Instructions for Evaluating Subject Matter Eligibility under 35 U.S.C. § 101, with respect to a product example of claimed judicial exception. For the Examiner's convenience, a copy of the Interim Examination instructions related to the Judicial Exception is attached. Claim 13 is patentable over the cited art for at least the same reasons as claim 12.

In view of the amendments and arguments set forth above, the above-identified application is in condition for allowance which action is respectfully requested.

Respectfully submitted,

RatnerPrestia

Lawrence E. Ashery, Reg. No. 34,515
Attorney for Applicant

LEA/kpc

Attachments: Interim Examination Instructions for Evaluating Subject Matter Eligibility Under 35 U.S.C. § 101
U.S. Patent Application Publication US 2001/0051936 A1 to Michalewicz

Dated: October 11, 2010

P.O. Box 980
Valley Forge, PA 19482
(610) 407-0700

The Director is hereby authorized to charge or credit Deposit Account No. **18-0350** for any additional fees, or any underpayment or credit for overpayment in connection herewith.

1042421

PRODUCT EXAMPLE: CLAIM 2
Judicial Exception Claimed

Claim 2. A machine for evaluating search results, comprising:

- a microprocessor coupled to a memory,
- wherein the microprocessor is programmed to evaluate search results by:
  - sorting the results into groups based on a first characteristic;
  - ranking the results based on a second characteristic using a mathematical formula [f]; and
  - comparing the ranked results to a predetermined list of desired results to evaluate the success of the search.

- Is the claim directed to a machine? (P1)
  - YES - it is a concrete thing, consisting of parts.
- Does it recite a judicial exception? (P3)
  - YES - the ranking step includes a mathematical algorithm.
- Is it directed to a practical application? (P4)
  - YES - evidenced by the tangible embodiment of the microprocessor for evaluating search results, which is a real world use.
- Is the claim directed to substantially all practical applications of the mathematical algorithm? (P5)
  - NO – the algorithm is limited to use in evaluating search results in the particular claimed machine that is programmed to perform certain steps. As there are other ways to use the algorithm, for example, with different programmed steps, not every use is covered by the claim.

➢ The claim is **eligible** (P6).

US 20010051936A1

(19) United States
(12) Patent Application Publication
Michalewicz
(10) Pub. No.: US 2001/0051936 A1
(43) Pub. Date: Dec. 13, 2001

(54) SYSTEM AND METHOD FOR DETERMINING AN OPTIMUM OR NEAR OPTIMUM SOLUTION TO A PROBLEM

(76) Inventor: Zbigniew Michalewicz, Charlotte, NC (US)

Correspondence Address:
McGuireWoods
1750 Tysons Boulevard, Suite 1800
Tysons Corner
McLean, VA 22102-4215 (US)

(21) Appl. No.: 09/837,194

(22) Filed: Apr. 19, 2001

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/198,643, filed on Apr. 20, 2000.

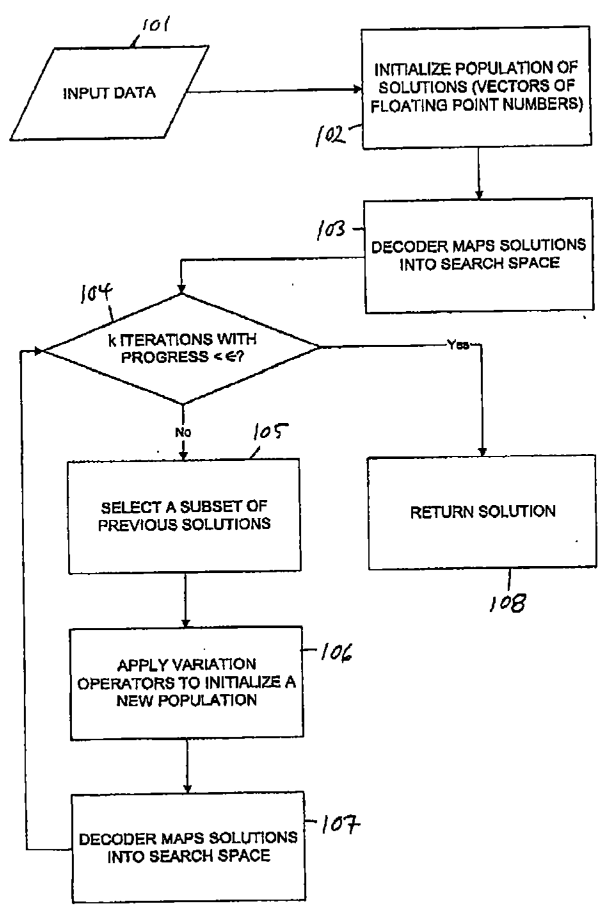
Publication Classification

(51) Int. Cl.[7] G06N 5/02; G06F 17/00

(52) U.S. Cl. 706/46

(57) ABSTRACT

A method and system for returning an optimum (or near-optimum) solution to a nonlinear programming problem. By specifying a precision coefficient, the user can influence the flexibility of the returned solution. A population of possible solutions is initialized based on input parameters defining the problem. The input parameters may include a minimum progress and a maximum number of iterations having less the minimum progress. The solutions are mapped into a search space that converts a constrained problem into an unconstrained problem. Through multiple iterations, a subset of solutions is selected from the population of solutions, and variation operators are applied to the subset of solutions so that a new population of solutions is initialized and then mapped. If a predetermined number of iterations has been reached, that is if the precision coefficient has been satisfied, the substantially optimum solution is selected from the new population of solutions. The system and method can be used to solve various types of real-world problems in the fields of engineering and operations research.



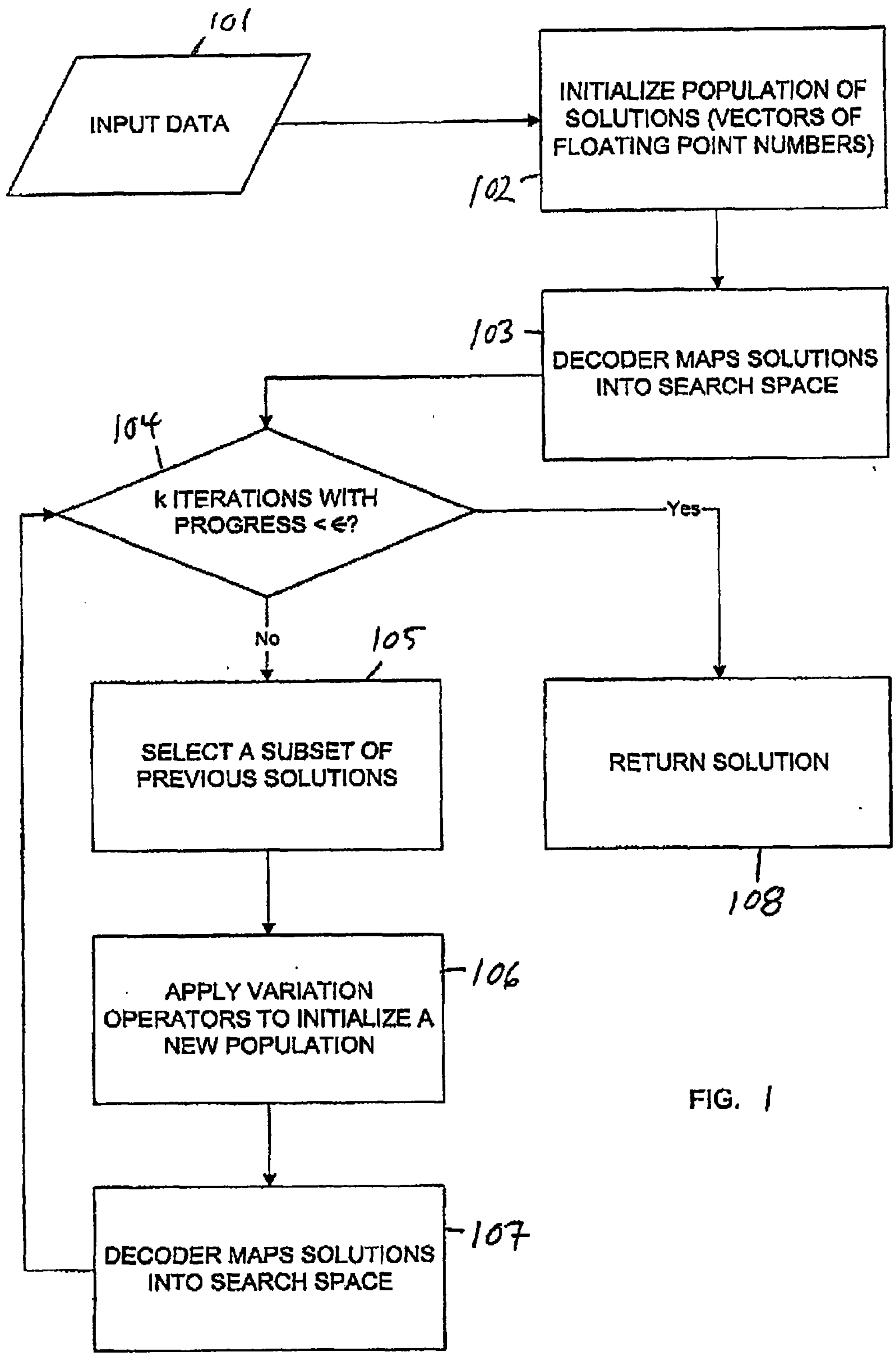

101
INPUT DATA
INITIALIZE POPULATION OF SOLUTIONS (VECTORS OF FLOATING POINT NUMBERS)
102
103
DECODER MAPS SOLUTIONS INTO SEARCH SPACE
104
k ITERATIONS WITH PROGRESS < ∈?
Yes
No
105
SELECT A SUBSET OF PREVIOUS SOLUTIONS
RETURN SOLUTION
108
APPLY VARIATION OPERATORS TO INITIALIZE A NEW POPULATION
106
DECODER MAPS SOLUTIONS INTO SEARCH SPACE
107

FIG. 1

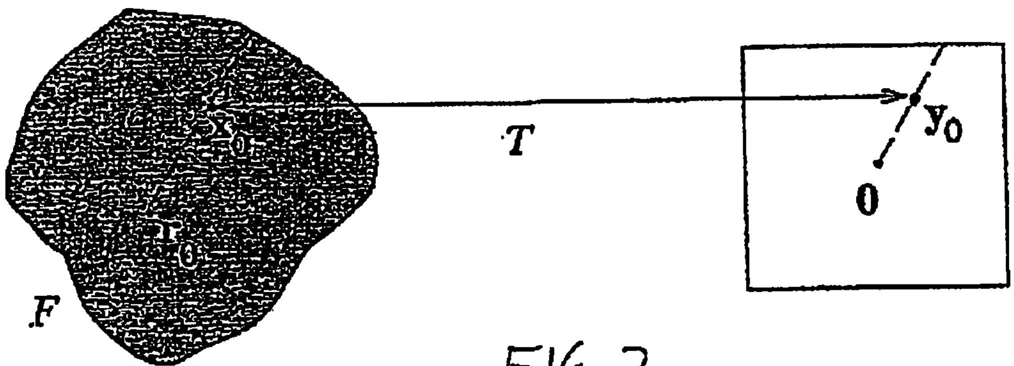


FIG. 2

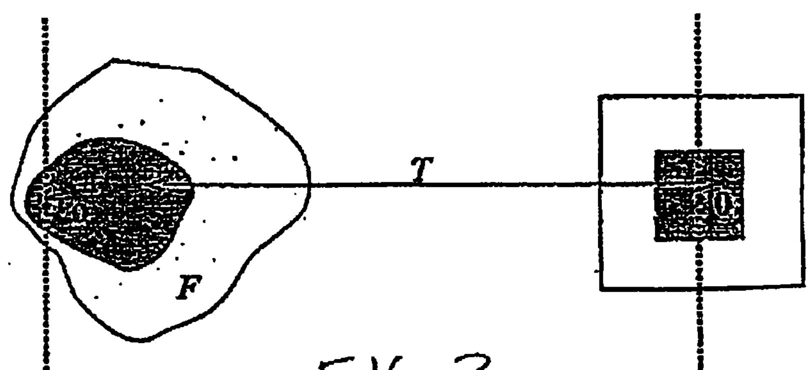


FIG. 3

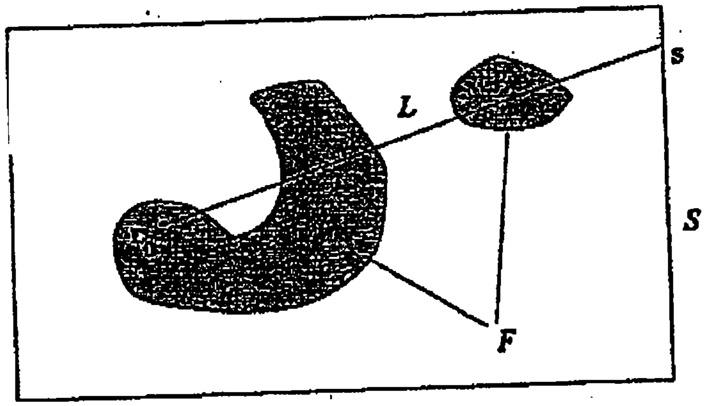

L
s
S
F

FIG. 4

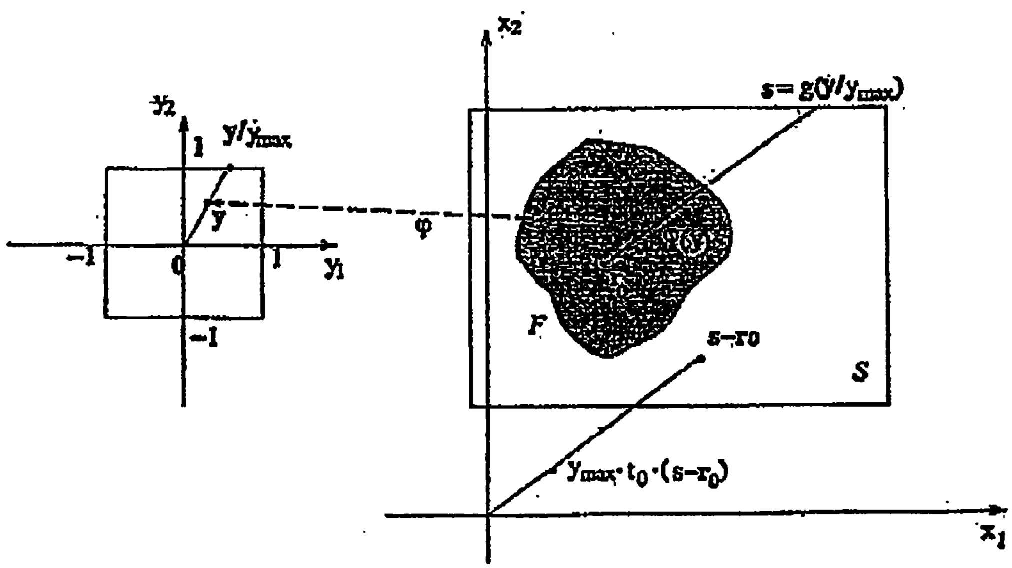

y_2
1
y/y_{max}
$\bar{y}$
−1
0
1
y_1
−1
φ
x_2
$s = g(\bar{y}/y_{max})$
F
$s-x_0$
S
$y_{max} \cdot t_0 \cdot (s-x_0)$
x_1

FIG. 5

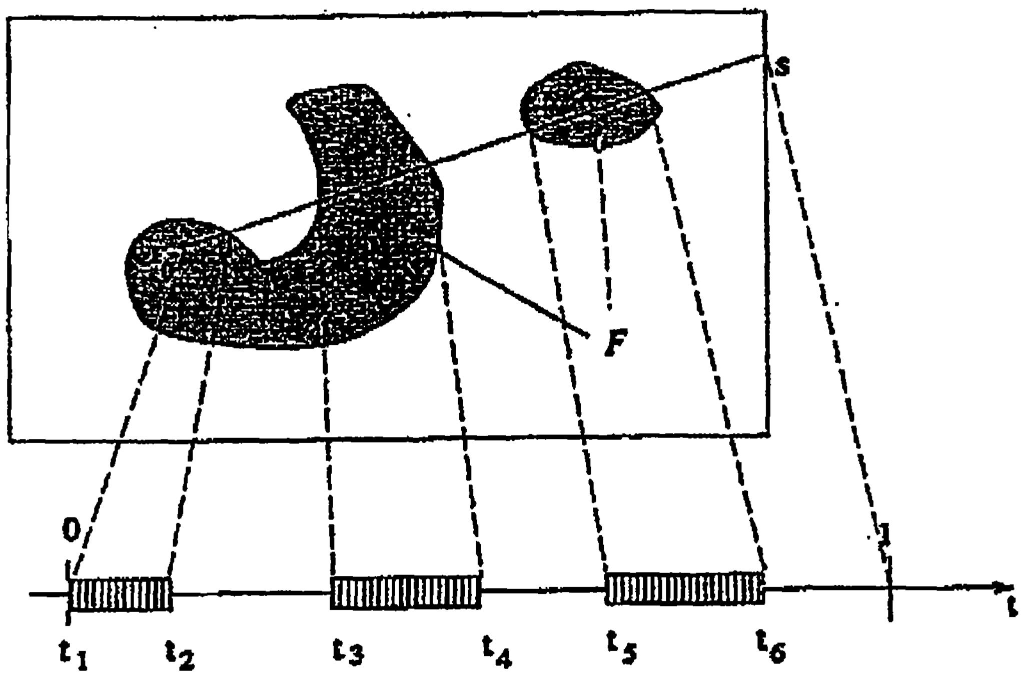
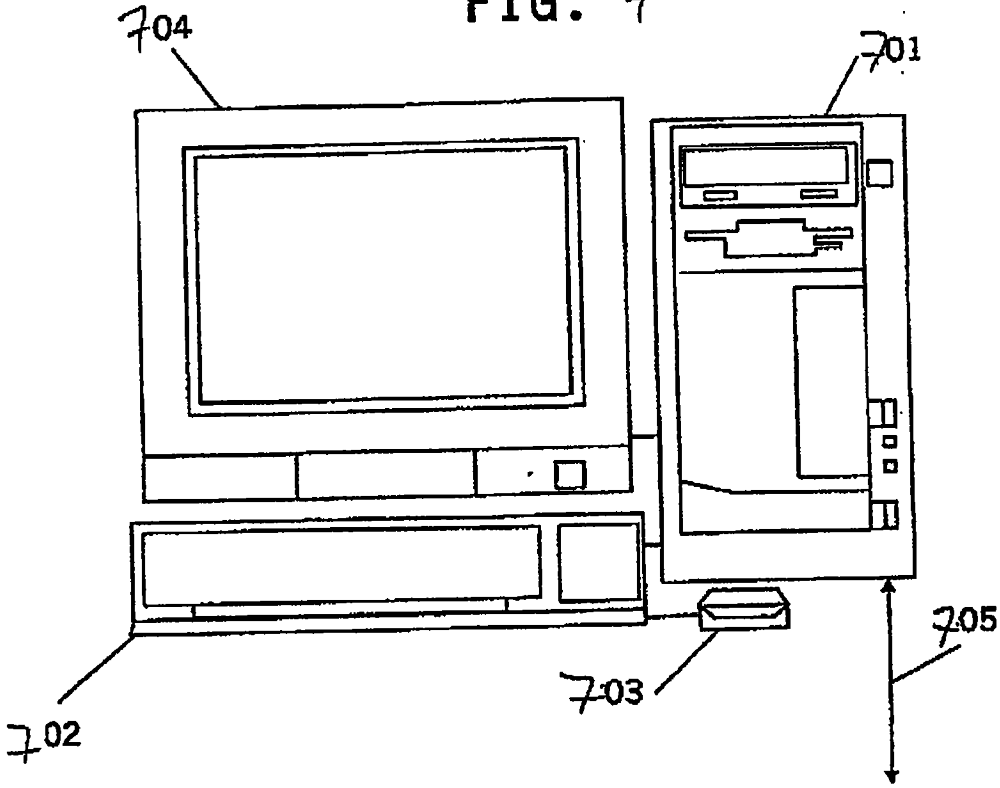


FIG. 6

FIG. 7



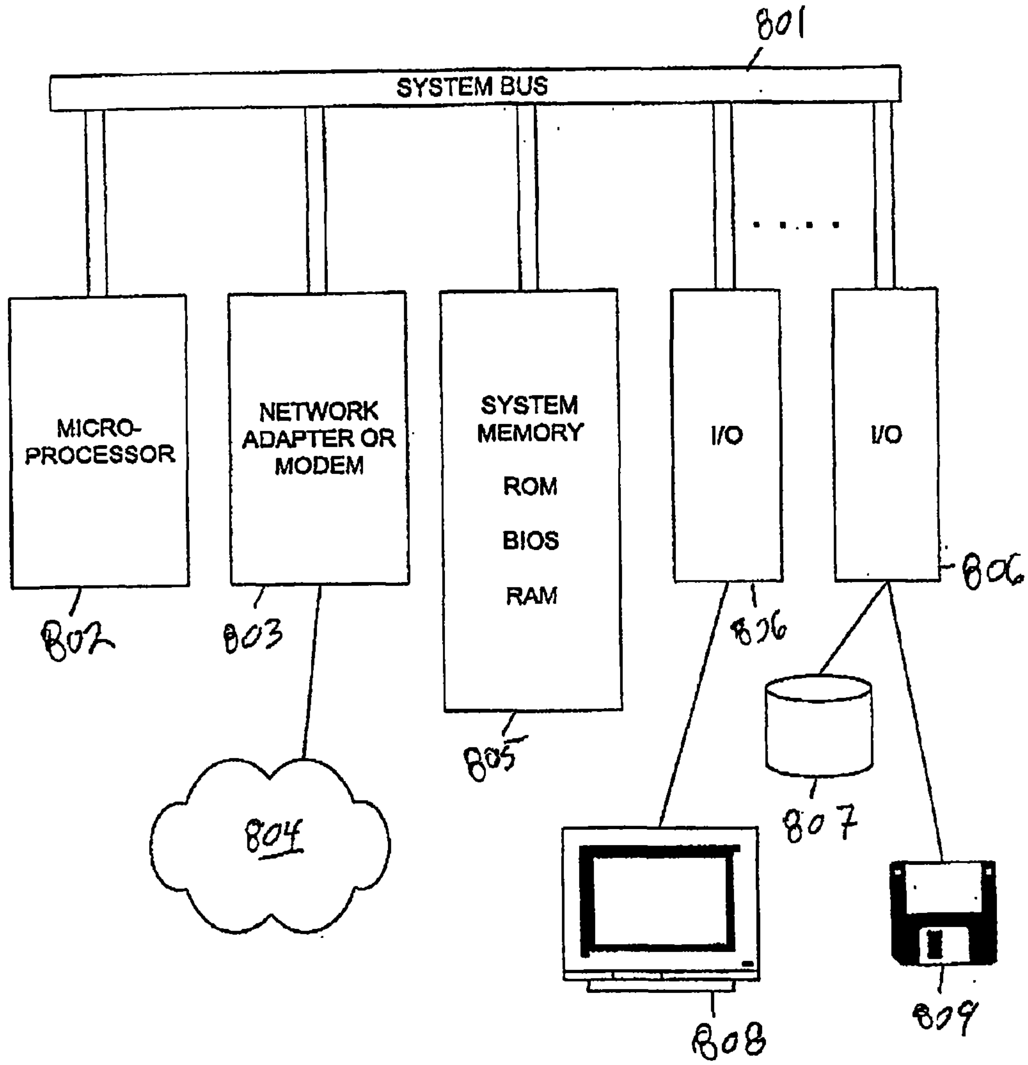

801
SYSTEM BUS
MICRO-
PROCESSOR
NETWORK
ADAPTER OR
MODEM
SYSTEM
MEMORY
ROM
BIOS
RAM
I/O
I/O
806
802
803
806
804
805
807
808
809

FIG. 8

SYSTEM AND METHOD FOR DETERMINING AN OPTIMUM OR NEAR OPTIMUM SOLUTION TO A PROBLEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Application Ser. No. 60/198,643, filed on Apr. 20, 2000, the entire contents of which are herein incorporated by reference.

BACKGROUND

[0002] 1. Field of the Invention

[0003] The present invention generally relates to the field of nonlinear programming and, more particularly, the present invention relates to a system and method implementing nonlinear programming techniques to determine an optimum or near-optimum solution to a real-world problem.

[0004] 2. Background Description

[0005] Nonlinear programming is a technique that can be used to solve problems that can be put into a specific mathematical form. Specifically, nonlinear programming problems are solved by seeking to minimize a scalar function of several variables subject to other functions that serve to limit or define the values of the variables. These other functions are typically called constraints. The entire mathematical space of possible solutions to a problem is called the search space and is usually denoted by the letter "S". The part of the search space in which the function to be minimized meets the constraints is called the feasibility space and is usually denoted by the letter "F".

[0006] Nonlinear programming is a difficult field, and often many complexities must be conquered in order to arrive at a solution or "optimum" to a nonlinear programming problem. For example, some problems exhibit local "optima"; that is, some problems have spurious solutions that merely satisfy the requirements of the derivatives of the functions. However, nonlinear programming can be a powerful tool to solve complex real-world problems, assuming a problem can be characterized or sampled to determine the proper functions and parameters to be used in the nonlinear program.

[0007] Due to the complexity of nonlinear programming techniques, computers are often used to implement a nonlinear program. It should be noted that the term "programming" as used in the phrase "nonlinear programming" refers to the planning of the necessary solution steps that is part of the process of solving a particular problem. This choice of name is incidental to the use of the terms "program" and "programming" in reference to the list of instructions that is used to control the operation of a modem computer system. Thus, the term "NLP program" for nonlinear programming software is not a redundancy.

[0008] Almost any type of problem can be characterized in a way that allows it to be solved with the help of NLP techniques. This is because any abstract task to be accomplished can be thought of as solving a problem. The process of solving such a problem can, in turn, be perceived as a search through a space of potential solutions. Since one usually seeks the best solution, this task can be characterized as an optimization process. However, nonlinear programming techniques are especially useful for solving complex engineering problems. These techniques can also be used to solve problems in the field of operations research (OR) which is a professional discipline that deals with the application of information technology for informed decision-making.

[0009] The majority of numerical optimization algorithms for nonlinear programming are based on some sort of local search principle; however, there is quite a diversity of these methods. Of course, then, classifying them neatly into separate categories is difficult because there are many different options. By example, some incorporate heuristics for generating successive points to evaluate, others use derivatives of the evaluation function, and still others are strictly local, being confined to a bounded region of the search space. But these numerical optimization algorithms all work with complete solutions and they all search the space of complete solutions. Most of these techniques make assumptions about the objective function or constraints of the problem (e.g., linear constraints, quadratic function, etc), and most of these techniques also use some type of penalty function to handle problem-specific constraints.

[0010] One of the many reasons that there are so many different approaches to nonlinear programming problems is that no single method has emerged as superior to all others. In general, it has been thought impossible to develop a deterministic method for finding the best global solution in many situations that would be better than an exhaustive search. There is thus a need for a method and system that can be used to find optimal or near-optimal solutions for almost any nonlinear programming problem. Ideally, the method and system should be able to handle both linear and nonlinear constraints.

SUMMARY

[0011] The present invention can be used to find an optimal (or near-optimal) solution to any nonlinear programming problem. The objective function need not be continuous or differentiable. The method and system according to the present invention will return the optimum (or near-optimum) solution which is feasible (i.e., it satisfies problem-specific constraints).

[0012] According to the method of the present invention, a population of possible solutions is initialized based on input parameters defining the problem. The input parameters may include, for example, a minimum progress and a maximum number of iterations having less the minimum progress (where the minimum progress may be the precision coefficient). The solutions are mapped into a search space by a decoder. For most problems the input parameters also include such features as, for example, all variables of the problem, the domains for the variables, the formula for the objective function, and the constraints (linear and nonlinear).

[0013] After mapping the problem into a search space (which converts the constrained problem into an unconstrained problem) the method of the present invention then proceeds by repeatedly selecting a subset of solutions from the population of solutions, applying variation operators to the subset of solutions so that a new population of solutions is initialized, and mapping the new population of solutions

into the search space. Finally, when termination condition is satisfied (e.g., the maximum number of iterations having less than the minimum progress has been reached, i.e., if the precision coefficient has been satisfied), the substantially optimum solution is selected from the new population of solutions. This solution can be supplied to a file for later retrieval, or supplied directly into another computerized process. The variation operators mentioned above include both unary and binary operators.

[0014] A computer software program or hardwired circuit can be used to implement the present invention. In the case of software, the program can be stored on media such as, for example, magnetic media (e.g., diskette, tape, or fixed disc) or optical media such as a CD-ROM. Additionally, the software can be supplied via the Internet or some other type of network. A workstation or personal computer that typically runs the software includes a plurality of input/output devices and a system unit that includes both hardware and software necessary to provide the tools to execute the method of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] **FIG. 1** shows a flowchart which illustrates the method of the present invention;

[0016] **FIG. 2** is a diagram illustrating how a space is mapped into a cube in order to initialize a search space according to the invention;

[0017] **FIG. 3** illustrates the influence of the location of a reference point on a transformation according to the invention;

[0018] **FIG. 4** shows a line segment of a non-convex space that follows from the mapping of the present invention;

[0019] **FIG. 5** is a diagram that illustrates mapping from a cube into a convex space according to the invention;

[0020] **FIG. 6** shows a line segment in a non-convex space and corresponding sub-intervals resulting from mapping which is implemented according to the present invention;

[0021] **FIG. 7** illustrates a workstation on which the present invention is implemented; and

[0022] **FIG. 8** illustrates further detail of an embodiment of hardware for implementing the system and method of the present invention.

DETAILED DESCRIPTION

[0023] The present invention is directed to optimally or near-optimally providing solutions to complex real-world problems which may be encountered in any number of situations. By way of illustration and not to limit the present invention in any manner, a type of problem that the system and method of the present invention may solve is a design engineering problem such as the design of an engine which is modeled by an array of parameters (e.g., 100 different variables) such as pressures, lengths, component type and the like. These parameters may be labeled $x_1, x_2, \ldots x_{100}$. In providing a solution to this and other problems, the present invention will minimize some very complex objective that is given as a formula of these 100 variables, or as a procedure to execute using these 100 variables.

[0024] Also, in this specific illustration there may also be problem-specific constraints. For example, the total of three dimensions (e.g., x_4, x_5, and x_6) of a particular part on the engine may have to be designed to stay between 10 and 15. This constraint may be modeled as a pair of linear constraints such that:

$$x_4+x_5+x_6 \geqq 10, \text{ and}$$

$$x_4+x_5+x_6 \leqq 15$$

[0025] Similarly, it is possible to have nonlinear constraints (e.g., a volume should stay within some limit). Thus, the problem can be specified by the objective function, the variables, their domains, and a set of constraints.

[0026] The general nonlinear programming (NLP) problem is to find x so as to:

$$\text{optimize } f(\vec{x}),\ \vec{x}(x_1, \ldots, x_1)\epsilon\mathfrak{R}^n,$$

[0027] where $\vec{x}\epsilon F\subseteq\mathfrak{R}^n$. The objective function "f" is defined on the search space $S\subseteq\mathfrak{R}^n$ and the set $F\subseteq S$ defines the feasible region. Usually, the search space S is defined as an n-dimensional rectangle in $\mathfrak{R}^n$ (domains of variables defined by their lower and upper bounds):

$$l(i)\leqq xi\leqq u(i),\ 1\leqq i\leqq n,$$

[0028] whereas the feasible region $F\subseteq S$ is defined by a set of m additional constraints ($m\geqq 0$):

$$g_j(\vec{x})\leqq 0, \text{ for } j=1, \ldots, q, \text{ and } h_j(\vec{x})=0, \text{ for } j=q+1, \ldots, m.$$

[0029] It is a common practice to replace the equation $h_j(\vec{x})=0$ with a set of inequalities, $h_j(\vec{x})\leqq\delta$ and $h_j(\vec{x})\geqq-\delta$ for some small $\delta>0$. Throughout the remaining portions of the disclosure, it is assumed that the above holds true.

[0030] Consequently, the set of constraints consists of m inequalities $g_j(\vec{x})\leqq 0$, for $j=1, \ldots, m$. After replacement of the equations with pairs of inequalities, the total number of inequality constraints is actually $q+2\cdot(m-q)=2m-q$. However, to simplify the notation, it is assumed there are m inequality constraints. At any point $\vec{x}\epsilon F$, the constraints g_j that satisfy $g_j(\vec{x})=0$ are called the active constraints at $\vec{x}$.

[0031] The NLP problem has often been thought of as intractable; that is, it is impossible to develop a deterministic method for the NLP in the global optimization category, which would be better than an exhaustive search. However, this makes room for the system and method of the present invention extended by some constraint-handling methods such as described herein in accordance with the present invention. The evolutionary method and system of the present invention uses specialized operators and a decoder. The decoder is based on the transformation of a constrained problem to an unconstrained problem via mapping. This method has numerous advantages, including, not requiring additional parameters, not having a need to evaluate or penalize infeasible solutions, and easiness of approaching a solution located at the edge of a feasible region.

[0032] As previously mentioned, specialized operators are used to implement the invention. These operators "assume" that the search space is convex. The domain D is defined by ranges of variables ($l_k\leqq x_k\leqq r_k$ for $k=1, \ldots n$) and by a set of constraints C. From the convexity of the set D it follows that for each point in the search space $(x_1, \ldots, x_n)\epsilon D$ there

exists a feasible range {left(k), right(k)} of a variable $x_k (1 \leq k \leq n)$, where other variables $x_i (i=1, \ldots, k-1, k+1, \ldots n)$ remain fixed. In other words, for a given $(x_1, \ldots, x_k, \ldots, x_n) \in D$:

$$y \in \{left(k), right(k)\} \text{ iff } (x_1, \ldots, x_{k-1}, y, x_{k+1}, \ldots, x_n) \in D,$$

[0033] where all x_i's $(i=1, \ldots, k-1, k+1, \ldots, n)$ remain constant. We assume also that the ranges {left(k), right(k)} can be efficiently computed.

[0034] If the set of constraints C is empty, then the search space $D=H_{k-1}{}^{n}\{l_k, r_k\}$ is convex; additionally left(k)=l_k, right(k)=r_k for k=1, ... n. Therefore, the operators constitute a valid set regardless of the presence of the constraints.

[0035] Several operators based on floating point representation are used with the invention. The first three are unary operators, each representing a category of mutation. The other three operators are binary operators, representing various types of crossovers. The operators are discussed below.

[0036] Uniform Mutation

[0037] This operator requires a single parent $\vec{x}$ and produces a single offspring $\vec{x}'$. The operator selects a random component $k \in (1, \ldots, n)$ of the vector $\vec{x}=(x_1, \ldots, x_k, \ldots, x_n)$ and produces $\vec{x}'=(x_1, \ldots, x'_k, \ldots, x_n)$, where x'_k is a random value (uniform probability distribution) from the range {left(k), right(k)}.

[0038] Boundary Mutation

[0039] This operator also requires a single parent $\vec{x}$ and produces a single offspring $\vec{x}'$. The operator is a variation of the uniform mutation with x'_k being either left(k) or right(k), each with equal probability. The operator is constructed for optimization problems where the optimal solution lies either on or near the boundary of the feasible search space. Consequently, if the set of constraints C is empty, and the bounds for variables are quite wide, the operator is a nuisance. But this operator can prove extremely useful in the presence of constraints.

[0040] Non-uniform Mutation

[0041] This is a unary operator responsible for the fine tuning capabilities of the system and method of the present invention. The operator is defined as follows. For a parent $\vec{x}$, if the element x_k was selected for this mutation, the result is $\vec{x}'=\{x_1, \ldots x'_k, \ldots, x_q\}$, where:

$$x'_k = \begin{cases} x_k + \Delta(t, \text{right}(k) - x_k) & \text{if a random binary digit is 0} \\ x_k - \Delta(t, x_k - \text{left}(k)) & \text{if a random binary digit is 1.} \end{cases}$$

[0042] The function $\Delta(t,y)$ returns a value in the range $[0,y]$ such that the probability of $\Delta(t,y)$ being close to 0 increases as t increases (t is the generation number). This property causes this operator to search the space uniformly initially (when t is small), and very locally at later stages. $\Delta(t,y)$ can be specified by the following function:

$$\Delta(t, y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b,$$

[0043] where r is a random number from [0..1], T is the maximal generation number, and b is a system parameter determining the degree of non-uniformity.

[0044] Arithmetical Crossover

[0045] This binary operator is defined as a linear combination of two vectors. If $\vec{x}_1$ and $\vec{x}_2$ are crossed, the resulting offspring are:

$$\vec{x}'_1 = a \cdot x_1 + (1-a) \cdot x_2 \text{ and } \vec{x}'_2 = a \cdot x_2 + (1-a) \cdot x_1.$$

[0046] This operator uses a random value $a \in [0..1]$, as it always guarantees closedness $(\vec{x}'_1, \vec{x}'_2 \in D)$.

[0047] Simple Crossover

[0048] This is a binary operator such that if $\vec{x}_1=(x_1, \ldots, x_n)$ and $\vec{x}_2=(y_1, \ldots, y_n)$ are crossed after the k-th position, the resulting offspring are:

$$\vec{x}'_1=(x_1, \ldots x_k, y_{k+1}, \ldots, y_n) \text{ and } \vec{x}'_2=(y_1, \ldots y_k, x_{k+1}, \ldots, x_n).$$

[0049] Such an operator may produce offspring outside the domain D. To avoid this, the present invention uses the property of the convex spaces stating that there exists $a \in [0,1]$ such that:

$$\vec{x}'_1+\{x_1, \ldots, x_k, y_{k+1} \cdot a + x_{k+1}(1-a), \ldots, y_n \cdot a + x \cdot (1-a)\}$$

[0050] and

$$\vec{x}'_2=\{y_1, \ldots, y_k, x_{k+1} \cdot a + y_{k+1} \cdot (1-a), \ldots, x_n \cdot a + y_n \cdot (1-a)\}$$

[0051] are feasible.

[0052] Heuristic Crossover

[0053] This operator is a unique crossover for the following reasons: (1) it uses values of the objective function in determining the direction of the search, (2) it produces only one offspring, and (3) it may produce no offspring at all. This operator generates a single offspring $\vec{x}_3$ from two parents, $\vec{x}_1$ and $\vec{x}_2$ according to the following rule:

$$\vec{x}_3 = r \cdot (\vec{x}_2 - \vec{x}_1) + \vec{x}_2,$$

[0054] where r is a random number between 0 and 1, and the parent $\vec{x}_2$ is no worse than $\vec{x}_1$, i.e., $f(\vec{x}_2) \geqq f(\vec{x}_1)$ for maximization problems and $f(\vec{x}_1) \leqq f(\vec{x}_1)$ for minimization problems.

[0055] It is possible for this operator to generate an offspring vector which is not feasible. In such a case another random value r is generated and another offspring is created. If after w attempts no new solution meeting the constraints is found, the operator stops and produces no offspring. The heuristic crossover contributes towards the precision of the solution found, where its major responsibilities are (1) fine local tuning and (2) searching in the promising direction.

[0056] However, it is necessary to be able to handle cases where the feasible search space is not convex. In order for

the present invention to be able to handle such cases, a decoder is used. In techniques based on decoders, a chromosome "gives instructions" on how to build a feasible solution. For example, a sequence of items for the classic knapsack problem can be interpreted as: "take an item if possible." Such an interpretation would lead always to a feasible solution.

[0057] Several factors should be taken into account while using a decoder. A decoder imposes a mapping T between a feasible solution and decoded solution. It is important that this mapping satisfies several conditions. First, for each solution $s\epsilon F$ there must be an encoded solution d. Also, each encoded solution d should correspond to a feasible solution s. All solutions in F should be represented by the same number of encodings d. Additionally, it is reasonable to expect that the transformation T is computationally fast and that it has a locality feature in the sense that small changes in the coded solution result in small changes in the solution, itself.

[0058] Now understanding the above, **FIG. 1** shows a flowchart illustrating the method of the invention using a decoder which meets all the above requirements and the variation operators as described above. It should be well understood by those of ordinary skill in the art that **FIG. 1** may equally represent a high level system block diagram of the present invention.

[0059] At step **101**, input data is organized into modules. These modules may be created manually, or created by another program or routine in the computer software that is implementing the invention. In embodiments, one module includes the number of variables, their domains, and all linear constraints. In embodiments, another module includes the objective function, while a third module includes all nonlinear constraints.

[0060] At step **102**, a population of solutions is initialized. That is, a number of potential solutions to the problem are generated by the method of the present invention. All solutions are vectors of floating point numbers. Each component of each vector is a number from the range [0..1]. At step **103**, the decoder of the present invention initially maps the solutions into a search space. It is noted that each individual solution is mapped into a feasible solution from the real search space. The mechanism of this mapping is further described below.

[0061] Steps **104** through **107** describe the iterations that take place after the initial mapping in order to reach a final solution to the problem. At step **104**, a termination condition is described. For example, if there have been "k" iterations with progress less than ϵ (the precision coefficient), the process stops and the current solution is returned at step **108**. Initially, there have been no iterations, so steps **105**-**107** are performed until there have been "k" iterations. At step **105**, a subset of solutions from the search space is selected according to a biased probability distribution, where better solutions have better chances for selection. One or more of the variation operators are applied to the subset at step **106** to arrive at a new, smaller population of solutions. The input file specifies the operators and their frequency. These new solutions are then mapped into the search space at step **107**, and the process repeats until the condition at step **104** is met and the best or most optimum solution is returned at step **108**. The returned best solution can be presented on a screen, stored in a file, or a numerical description of the solution can serve as input to another program or computerized process.

[0062] The mapping process and the decoder can be most readily understood by examining a nonlinear programming process. **FIG. 2** shows a one-to-one mapping between an arbitrary convex feasible search space F and an n-dimensional cube $[-1,1]^n$. An arbitrary (different than $\vec{0}$) point:

$$\vec{y}_0=(y_{0,1}, \ldots, y_{0,n})\epsilon[-1,1]^n$$

[0063] defines a line segment from the vector $\vec{0}$ to the boundary of the cube. This segment is described by:

$$y_i=y_{0,i}\cdot t, \text{ for } i=1, \ldots, n \text{ where}$$

[0064] t varies from 0 to $t_{max}=1/\max\{|y_{0,1}|, \ldots, |y_{0,n}|\}$. For t=0, and for $t=t_{max}$, $\vec{y}=(y_{0,1}t_{max}, \ldots, y_{0,n}t_{max})$ a boundary condition of the $[-1,1]^n$ cube is represented. Consequently, the corresponding feasible point $\vec{x}_0\epsilon F$ is defined as:

$$\vec{x}_0=\vec{r}_0+\vec{y}_0\tau,$$

[0065] where $\tau=\tau_{max}/t_{max}$, and τ_{max} is determined with arbitrary precision by a binary search procedure such that

$$\vec{r}_0+\vec{y}_0\tau_{max}$$

[0066] is a boundary point of the feasible search space F. This mapping satisfies all the previously mentioned requirements for the decoder.

[0067] Apart from being one-to-one, the transformation is fast and has a locality feature. The corresponding feasible point $\vec{x}_0\epsilon F$ is defined with respect to some reference point $\vec{r}_0$. Such a reference point is an arbitrary internal point of the convex set F. Note that convexity of the feasible search space is not necessary, but it is sufficient to assume the existence of the reference point $\vec{r}_0$ such that every line segment originating in $\vec{r}_0$ intersects the boundary of F at precisely one point. This requirement is satisfied for any convex set F.

[0068] This approach may be extended by the additional method of iterative solution improvement according to the present invention. The iterative solution improvement of the present invention is based on the relationship between the location of the reference point and the efficiency of the proposed approach. It is clear that the location of the reference point $\vec{r}_0$ has an influence on "deformation" of the domain of optimized function. The present invention optimized some other function which is topologically equivalent to the original function. For example, consider the case, shown in **FIG. 3**, where the reference point is located near the edge of the feasible region F. It is easy to notice a strong irregularity of transformation T. The part of the cube $[-1,1]^2$, which is on the left side of the vertical line, is transformed into a much smaller part of the set F than the part on the right side of this line.

[0069] According to the above considerations, it is profitable to localize the reference point in the neighborhood of the expected optimum, if this optimum is close to the edge of the set F. In such case the area between the edge of F and the reference point $\vec{r}_0$ is explored more precisely.

[0070] In the case of lack of information about approximate localization of the solution, the reference point is placed close to the geometrical center of the set F. This can be done by sampling set F and setting:

$$\vec{r}_0=1/k\Sigma_{i=1}^{k}\vec{x}_i,$$

[0071] where $\vec{x}_i$ consists of samples from F. It is also possible to take advantage of the mentioned effect for the purpose of iterative improvement of the best-found solution. To obtain this effect it is necessary to repeat the optimization process with a new reference point $\vec{r}'_0$ which is located on a line segment between the current reference point $\vec{r}_0$ and the best solution $\vec{b}$ found to this point:

$$\vec{r}'_0=t\cdot\vec{r}_0+(1-t)\cdot\vec{b},$$

[0072] where $t\in(0,1]$ is close to zero. This change of the location of the reference point causes the found optimum to be explored more precisely in the next iteration in the neighborhood in comparison with the remaining part of the feasible region. Experiments have show that such a method usually provides good results for problems with optimal solutions localized on the edge of the feasible region.

[0073] The approach of the present invention can be also extended to handle non-convex search spaces (the original nonlinear programming problem). That is, the proposed technique of the present invention can handle arbitrary constraints for numerical optimization problems. The task is to develop a mapping ϕ, which transforms the n-dimensional cube $[-1,1]^n$ into the feasible region F of the problem. Note, that F need not be convex; it might be concave or even can consist of disjoint (non-convex) regions.

[0074] As shown in **FIG. 4**, this mapping ϕ is more complex than T defined earlier. Note that in **FIG. 4** any line segment L which originates at a reference point $\vec{r}_0\in F$ may intersect a boundary of the feasible search space F in more than just one point.

[0075] Because of the complexity of this mapping, it may be necessary to take into account the domains of the variables. First, an additional one-to-one mapping g between the cube $[-1,1]^n$ and the search space S is defined (the search space S is defined as a Cartesian product of domains of all problem variables). Then the mapping g: $[-1,1]^n\rightarrow S$ can be defined as:

$$g(\vec{y})=\vec{x},$$

[0076] where

$$x_i = y_i\frac{u(i)-l(i)}{2}+\frac{u(i)+l(i)}{2},\ \text{for } i=1,\ \ldots,n$$

[0077] Indeed, for $y_i=-1$ the corresponding $x_i=l(i)$, and for $y_i=1$ the corresponding $x_i=u(i)$.

[0078] A line segment L between any reference point $\vec{r}_0\in F$ and a point $\vec{s}$ at the boundary of the search space S, is defined as:

$$L(\vec{r}_0,\vec{s})=\vec{r}_0+t\cdot(\vec{s}+\vec{r}_0),\ \text{for } 0\leq t\leq 1.$$

[0079] If the feasible search space F is convex, then the above line segment intersects the boundary of F in precisely one point, for some $t_0\in[0,1]$. Consequently, for convex feasible search spaces F, it is possible to establish a one-to-one mapping $\phi:[-1,1]^n$ as follows:

$$\varphi(\vec{r})=\begin{cases}\vec{r}_0+y_{max}\cdot t_0\cdot(g(\vec{y}/y_{max})-\vec{r}_0) & \text{if } \vec{y}\neq\vec{0}\\ \vec{r}_0 & \text{if } y=\vec{0}\end{cases}$$

[0080] where $\vec{r}_0\in F$ is a reference point, and $y_{max}=\max_{i=1}^{n}|y_i|$. **FIG. 5** illustrates the transformation. That is, **FIG. 5** shows a mapping ϕ from the cube $[-1,1]^n$ into the convex space F (two-dimensional case), with the particular steps of the transformation.

[0081] Returning now to the general case of arbitrary constraints (i.e., non-convex feasible search spaces F), consider an arbitrary point $y\in[-1,1]^n$ and a reference point, $\vec{r}_0\in F$. A line segment L between the reference point $\vec{r}_0$ and the point $\vec{s}=g(\vec{y}/y_{max})$ at the boundary of the search space S, is defined as before:

$$L(\vec{r}_0,\vec{s})=\vec{r}_0+t\cdot(\vec{s}-\vec{r}_0),\ \text{for } 0\leq t\leq 1,$$

[0082] However, the line segment may intersect the boundary of F in many points as shown in **FIG. 4**. In other words, instead of a single interval of feasibility $[0,t_0]$ for convex search spaces, there may be several intervals of feasibility:

$$[t_1,t_2],\ \ldots,\ [t_{2k-1},t_{2k}].$$

[0083] It is assumed that there are altogether k subintervals of feasibility for such a line segment and t_i's mark their limits. **FIG. 6** shows a line segment in a non-convex space F and corresponding intervals for a two-dimensional case. As shown in **FIG. 6**:

$$t_1=0, t_i<t_{i+1},\ \text{for } i=1,\ \ldots,\ 2k-1,\ \text{and } t_{2k}\leq 1.$$

[0084] Thus, it is necessary to introduce an additional mapping γ, which transforms interval $[0,1]$ into the sum of intervals $[t_{2i-1},t_{2i}]$. However, such a mapping γ rather between $[0,1]$ and the sum of intervals $(t_{2i-1},t_{2i}]$ as follows:

$$\gamma:(0,1]\rightarrow\cup_{i=1}^{k}(t_{2i-1},t_{2i}].$$

[0085] Note that, due to this change, the left boundary point is lost. This is not a serious problem, since the lost points can be approached with arbitrary precision. However, there are important benefits to this definition. It is possible to "glue together" intervals which are open at one end and closed at another end. Additionally, such a mapping is one-to-one. There are many alternatives for defining such a mapping. For example, a reverse mapping:

$$\delta:\cup_{i=1}^{k}(t_{2i-1},t_{2i}]\rightarrow(0,1]$$

[0086] can be defined as follows:

$$\delta(t)=(t-t_{2i-1}+\Sigma_{j=1}^{i-1}d_j)/d,$$

[0087] where $d_j=t_{2j}-t_{2j-1}$, $d=\Sigma_{j=1}^{k}d_j$, and $t_{2i-1}<t\leq t_{2i}$. The mapping y is reverse of δ:

$$\gamma(a) = t_{2j-1} + d_j \frac{a - \delta(t_{2j-1})}{\delta(t_{2j}) - \delta(t_{2j-1})},$$

[0088] where j is the smallest index such that $a \leq \delta(t_{2j})$.

[0089] From the above, the general decoder mapping ϕ is defined which is used as shown in **FIG. 1** for the transformation of constrained optimization problem to an unconstrained optimization problem for every feasible set F. The mapping ϕ is given by the formula:

$$\varphi(\vec{y}) = \begin{cases} \vec{r}_0 + t_0 \cdot (g(\vec{y}/y_{max}) - \vec{r}_0) & \text{if } \vec{y} \neq \vec{0}, \\ \vec{r}_0 & \text{if } y = \vec{0}, \end{cases}$$

[0090] where $\vec{r}_0 \in F$ is a reference point, $y_{max} = \max_{i=1}^{n} |y_i|$, and $t_0 = \gamma(|y_{max}|)$.

[0091] Finally, it is necessary to consider a method of finding points of intersection t_i as shown in **FIG. 6**. This is relatively easy for convex sets, since there is only one point of intersection. For non-convex sets, however, the problem is more complex.

[0092] In the embodiments of the invention, the following approach has been used to find the points of intersection for the non-convex sets. Consider any boundary point $\vec{s}$ of S and the line segment L determined by this point and a reference point $\vec{r}_0 \in F$. There are m constraints $g_i(\vec{x}) \leq 0$ and each of them can be represented as a function β_i of one independent variable t for a fixed reference point $\vec{r}_0 \in F$ and the boundary point $\vec{s}$ of S:

$$\beta_i(t) = g_i(L(\vec{r}_0, \vec{s})) = g_i(\vec{r}_0 + t \cdot (\vec{s} - \vec{r})), \text{ for } 0 \leq t \leq 1 \text{ and } i = 1, \ldots, m.$$

[0093] As stated earlier, the feasible region need not be convex so it may have more than one point of intersection of the segment L with the boundaries of the set F. Therefore, the interval [0,1] is partitioned into v subintervals $[v_{j-1}, v]$, where:

$$v_j - v_{j-1} = 1/v (1 \leq j \leq v),$$

[0094] so that equations $\beta_i(t)=0$ have, at most, one solution in every subinterval. The density v of the partition is adjusted experimentally. For all cases discussed in this disclosure v=20. In this case the points of intersection can be determined by a binary search. Once the intersection points between a line segment L and all constraints $g_i(\vec{x}) \leq 0$ are known, one can then determine intersection points between this line segment L and the boundary of the feasible set F. The flexibility of the solution is achieved by evaluating a solution in a particular way. That is, several solutions in the neighborhood of the current solution, as determined by the precision coefficient, are evaluated and averaged. The computational method handles both linear and nonlinear constraints, and this is capable of handling convex and non-convex feasible search spaces in an efficient manner in accordance with the method and system of the present invention.

[0095] As previously mentioned, it is convenient to execute the method described above on a computer system which has been programmed with appropriate software. **FIG. 7** illustrates a workstation on which the method of the present invention can be executed. Input/output (I/O) devices such as keyboard **702**, mouse **703** and display **704** are used by an operator to provide input and view information related to the operation of the invention. A system unit **701** is connected to all of the I/O devices and contains memory, media devices, and a central processing unit (CPU), all of which together may execute the method of the present invention. These devices in combination with the appropriate software are the means for carrying out the various steps involved in implementing the method of the present invention.

[0096] As previously mentioned, appropriate computer program code in combination with the appropriate hardware may be used to implement the method of the present invention invention. This computer program code is often stored on storage media such as a diskette, hard disk, CD-ROM, DVD-ROM or tape. The media can also be a memory storage device or collection of memory storage devices such a read-only memory (ROM) or random access memory (RAM). Additionally, the computer program code can be transferred to a workstation over the Internet or some other type of network. The method of the present invention can equally be hardwired into a circuit or computer implementing the steps of the present invention.

[0097] **FIG. 8** illustrates further detail of the system unit for the computer system shown in **FIG. 7**. The system is controlled by microprocessor **802**, which serves as the CPU for the system. System memory **805** is typically divided into multiple types of memory or memory areas, such as read-only memory (ROM), random-access memory (RAM) and others. If the workstation is an IBM compatible personal computer, for example, the system memory also contains a basic input/output system (BIOS). A plurality of general input/output (I/O) devices **806** such as a keyboard or a mouse are connected to various devices including a fixed disk **807**, a diskette drive **809** and a display **808**. The system may include another I/O device, a network adapter or modem, shown at **803**, for connection to a network **804**. This network connection may be used to download the software implementing the present invention for execution on the computer system. A system bus **801** interconnects the major components **802, 803, 805** and **806** of **FIG. 8**.

[0098] It should be noted that the system as shown in **FIGS. 7 and 8** is meant as an illustrative example only and should not be considered as a limiting factor in determining the scope of the present invention. For example, the present invention may be implemented on numerous types of general-purpose computer systems running operating systems such as Windows™ by Microsoft and various versions of UNIX and the like.

EXAMPLE OF USE

[0099] The present invention is particularly useful in workflow management problems, process problems, and engineering problems. By way of illustrative example, assume that the optimization model of a particular engineering problem is as follows:

[0100] Minimize

$$0 \leq 85.334407 + 0.006858\alpha_2 x_5 + 0.0006262 x_1 x_1 - 0.0022053 x_3 x_5 \leq 92$$

$$90 \leq 80.51249 + 0.0071317 x_2 x_3 + 0.0029955 x_1 x_2 + 0.0021813 x_3^2 \leq 110$$

$$20 \leq 9.300961 + 0.0047026 x_3 x_3 + 0.0012547 x_1 x_3 + 0.0019035 x_3 x_4 \leq 25,$$

[0101] For this particular function, the optimum solution is $(\vec{x})$=(78.0, 33.0, 29.995, 45.0, 36.776), with $F(\vec{x})=-30665.5$. Two constraints (upper bound of the first inequality and the lower bound of the third inequality) are active at the optimum. Note, however, that for most real problems this is not the case, i.e., neither the optimum solution nor the number of active constraints is known. The only reason for selecting the function F, as an example, is to underline the quality of the present invention.

[0102] At this stage, the system and method of the present invention can be used to find the best solution. The user then sets some parameters of the system such as, for example, population size, frequencies of operators, termination conditions (e.g., 5,000 generations) and the like. The system and method of the present invention then determines a feasible point (by random sampling of the search space) which will take a role of the reference point $\vec{r}_0$ (i.e., the first randomly generated feasible point was accepted as a reference point). Utilizing the above discussion, the present invention finds a solution of value −30664.5, which is a 0.0033 of one percent error. This is the optimum solution which is provided by the present invention.

[0103] It cannot be overemphasized that the practical applications of the present invention are almost unlimited. For example, the present invention can provide solutions to:

[0104] structural design systems;

[0105] flaw detection in engineered structures;

[0106] multiprocessor scheduling in computer networks;

[0107] physical design of integrated circuits;

[0108] scheduling activities for an array of different, diverse systems;

[0109] radar imaging; and

[0110] mass customization, to name just a few.

[0111] While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims. The following claims are in no way intended to limit the scope of the invention to specific embodiments.

1. A method of finding a substantially optimal solution to a constrained problem, the method comprising the steps of:

initializing a population of possible solutions based on input parameters defining a problem;

mapping the population of possible solutions into a search space;

selecting a subset of solutions from the population of possible solutions;

applying at least one variation operator to the subset of solutions in order to provide a new population of solutions;

mapping the new population of solutions into the search space;

repeating the selecting, applying and mapping the new population of solutions steps until a termination condition is satisfied;

selecting the substantially optimum solution from the new population of solutions.

2. The method of claim 1, wherein the termination condition is one of the input parameters.

3. The method of claim 2, wherein the termination condition is based on a minimum progress and a maximum number of iterations having less the minimum progress.

4. The method of claim 3, wherein

the selecting the subset of solutions from the population of solutions step is performed when the maximum number of iterations having less than the minimum progress has not been reached; and

the selecting of the substantially optimum solution step is performed when the maximum number of iterations having less than the minimum progress has been reached.

5. The method of claim 1, wherein the selecting the substantially optimum solution step is performed after the repeating step.

6. The method of claim 1, further comprising organizing input data into modules prior to the initializing step, the optimum solution being based on the input data.

7. The method of claim 6, wherein the modules are separated into a plurality of modules, wherein:

a first of the plurality of modules including a number of variables, domains and linear constraints associated with the input data;

a second of the plurality of modules includes an objective function associated with the input data; and

a third of the plurality of modules includes nonlinear constraints associated with the input data.

8. The method of claim 1, wherein the mapping the population of possible solutions into a search space converts the constrained problem into an unconstrained problem.

9. The method of claim 1, wherein the at least one variation operator is two or more variation operators.

10. The method of claim 1 wherein the at least one variation operator includes both unary and binary operators.

11. The method of claim 10, wherein the unary and binary operators are selected from the group of a uniform mutation operator, boundary mutation operator, non-uniform mutation operator, arithmetical crossover operator, simple crossover operator and heuristic crossover operator

12. The method of claim 1, wherein the optimum solution is displayed to a user.

13. The method of claim 1, wherein the selecting a subset of solutions from the population of possible solutions includes the step of locating a substantial geometric center of the population of possible solutions.

14. A method of finding a substantially optimal solution to a constrained problem, the method comprising the steps of:

initializing a population of solutions based on input parameters defining a problem, the input parameters including a minimum progress and a maximum number of iterations having less the minimum progress;

mapping the population of solutions into a search space so that the constrained problem is converted into an unconstrained problem;

selecting a subset of solutions from the population of solutions if the maximum number of iterations having less than the minimum progress has not been reached;

applying variation operators to the subset of solutions so that a new population of solutions is initialized if the subset of solutions has been selected;

mapping the new population of solutions into the search space if the new population of solutions has been initialized; and

selecting the substantially optimum solution from the new population of solutions if the maximum number of iterations having less than the minimum progress has been reached.

15. The method of claim 14, wherein the variation operators include both unary and binary operators.

16. An apparatus for finding a substantially optimal solution to a constrained problem, the apparatus comprising:

means for mapping a population of solutions into a search space so that the constrained problem is converted to an unconstrained problem;

means for creating an initial population of solutions based on input parameters defining the problem;

means for iteratively selecting a subset of solutions from a population of solutions;

means for iteratively applying at least one variation operator to the subset of solutions in order to provide a new population of solutions; and

means for selecting the substantially optimum solution from the new population of solutions after a termination condition is satisfied.

17. The apparatus of claim 16, wherein the termination condition is an input parameter which is based on a minimum progress and a maximum number of iterations having the minimum progress.

18. The apparatus of claim 17, further comprising means for determining if the predetermined maximum iterations has been reached, the predetermined maximum iterations is equal to the maximum number of iterations having less than the minimum progress.

19. A computer program product for enabling a computer system to find a substantially optimal solution to a constrained problem, the computer program product including a medium with a computer program embodied thereon, the computer program comprising:

computer program code for mapping a population of solutions into a search space so that the constrained problem is converted into an unconstrained problem;

computer program code for creating an initial population of solutions based on input parameters defining the problem, the input parameters including a minimum progress and a maximum number of iterations having the minimum progress;

computer program code for selecting a subset of solutions from a population of solutions;

computer program code for applying variation operators to the subset of solutions so that a new population of solutions is initialized;

computer program code for determining if the maximum number of iterations having less than the minimum progress has been reached; and

computer program code for selecting the substantially optimum solution from the new population of solutions.

20. The computer program product of claim 19, wherein the variation operators include both unary and binary operators.

21. A programmed computer system which is operable to find a substantially optimal solution to a constrained problem by performing the steps of:

initializing a population of solutions based on input parameters defining the problem, the input parameters including a minimum progress and a maximum number of iterations having the minimum progress;

mapping the population of solutions into a search space so that the constrained problem is converted into an unconstrained problem;

selecting a subset of solutions from the population of solutions if the maximum number of iterations having less than the minimum progress has not been reached;

applying variation operators to the subset of solutions so that a new population of solutions is initialized if the subset of solutions has been selected;

mapping the new population of solutions into the search space if the new population of solutions has been initialized; and

selecting the substantially optimum solution from the new population of solutions if the maximum number of iterations having less than the minimum progress has been reached.

* * * * *